

## BAB 5 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dilakukan implementasi sistem dan pengujian berdasarkan perancangan yang dibuat pada bab sebelumnya dan dijelaskan proses pengambilan data kemudian akan dibahas pada bab selanjutnya.

### 5.1 Pembangunan Testbed



**Gambar 5.1 Implementasi Sistem**

Pada Gambar 5.1 menunjukkan bahwa *middleware* dapat dibangun pada raspberry pi zero dengan menggunakan tambahan usb port menggunakan USB Hub sehingga *middleware* nantinya dapat memiliki dua *interface* jaringan yaitu wlan0 dengan menggunakan *wireless adapter* sebagai *access point* dan eth0 dengan menggunakan USB to LAN untuk terhubung ke data center sehingga *middleware* dapat mengirimkan data ke aplikasi. Setelah sistem berhasil diimplementasikan dilakukan pengujian testbed untuk mengetahui bahwa rancangan topologi jaringan sudah berjalan sesuai dengan harapan dan dapat saling terhubung satu dengan yang lainnya. Pada Gambar 5.2 menunjukkan bahwa *publisher* CoAP berhasil mengirim data suhu dan kelembapan setiap 30 detik sekali dengan melakukan "POST" data dengan topik home/kitchen ke alamat `coap://192.168.43.1/r/home/kitchen` melalui port 5683 dan *publisher* MQTT berhasil mengirim data suhu dan kelembapan setiap 30 detik sekali dengan melakukan "PUBLISH" data dengan topik home/barrack ke alamat `mqtt://192.168.43.1` melalui port 1883.

```
pi@raspberrypi-zero: ~
[STREAMING] Now streaming realtime logs for [all] processes
1|qoap | 1/4/2018 12:52:53 PM COAP - Incoming POST request from 192.
168.43.31 for home/kitchen
1|qoap | 1/4/2018 12:52:58 PM MQTT - Client 533509 publish a message
to home/barrack
1|qoap | 1/4/2018 12:53:23 PM COAP - Incoming POST request from 192.
168.43.31 for home/kitchen
1|qoap | 1/4/2018 12:53:28 PM MQTT - Client 533509 publish a message
to home/barrack
1|qoap | 1/4/2018 12:53:53 PM COAP - Incoming POST request from 192.
168.43.31 for home/kitchen
1|qoap | 1/4/2018 12:53:58 PM MQTT - Client 533509 publish a message
to home/barrack
1|qoap | 1/4/2018 12:54:23 PM COAP - Incoming POST request from 192.
168.43.31 for home/kitchen
1|qoap | 1/4/2018 12:54:53 PM COAP - Incoming POST request from 192.
168.43.31 for home/kitchen
1|qoap | 1/4/2018 12:54:58 PM MQTT - Client 533509 publish a message
to home/barrack
1|qoap | 1/4/2018 12:55:23 PM COAP - Incoming POST request from 192.
168.43.31 for home/kitchen
1|qoap | 1/4/2018 12:55:28 PM MQTT - Client 533509 publish a message
to home/barrack
```

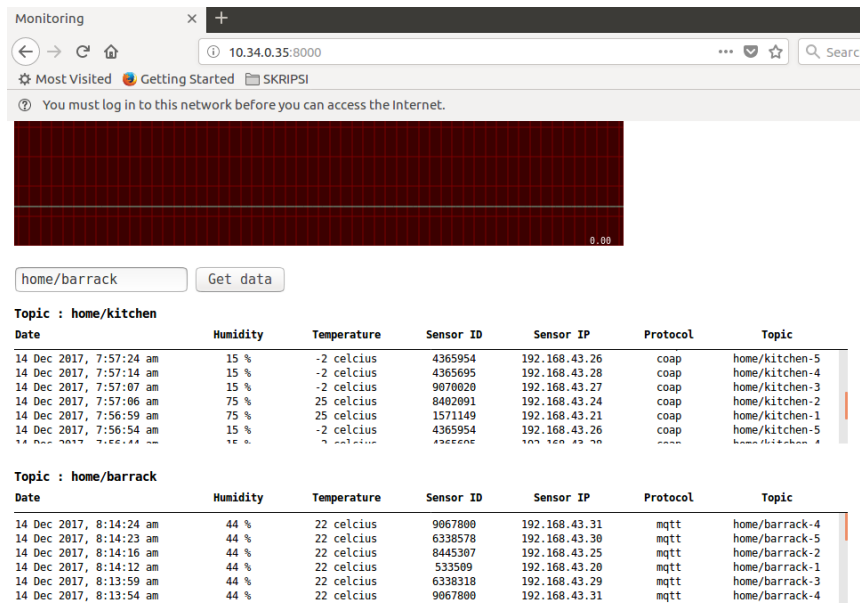
**Gambar 5.2 Data Publisher**

Untuk mengetahui apakah *middleware* telah terhubung dengan data center maka dilakukan tes ping ke alamat IP data center yaitu 10.34.0.35. Pada Gambar 5.3 menunjukkan bahwa *middleware* telah terhubung dengan data center dan dapat melakukan pertukaran data.

```
pi@raspberrypi-zero:~ $ ping 10.34.0.35
PING 10.34.0.35 (10.34.0.35) 56(84) bytes of data.
64 bytes from 10.34.0.35: icmp_seq=1 ttl=64 time=1.58 ms
64 bytes from 10.34.0.35: icmp_seq=2 ttl=64 time=1.25 ms
64 bytes from 10.34.0.35: icmp_seq=3 ttl=64 time=1.22 ms
64 bytes from 10.34.0.35: icmp_seq=4 ttl=64 time=1.24 ms
64 bytes from 10.34.0.35: icmp_seq=5 ttl=64 time=1.33 ms
64 bytes from 10.34.0.35: icmp_seq=6 ttl=64 time=1.28 ms
64 bytes from 10.34.0.35: icmp_seq=7 ttl=64 time=1.28 ms
64 bytes from 10.34.0.35: icmp_seq=8 ttl=64 time=1.48 ms
64 bytes from 10.34.0.35: icmp_seq=9 ttl=64 time=1.44 ms
64 bytes from 10.34.0.35: icmp_seq=10 ttl=64 time=1.41 ms
64 bytes from 10.34.0.35: icmp_seq=11 ttl=64 time=1.49 ms
64 bytes from 10.34.0.35: icmp_seq=12 ttl=64 time=1.34 ms
64 bytes from 10.34.0.35: icmp_seq=13 ttl=64 time=1.40 ms
^C
--- 10.34.0.35 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12021ms
rtt min/avg/max/mdev = 1.221/1.368/1.585/0.117 ms
```

**Gambar 5.3 Tes Ping ke Datacenter**

Setelah *middleware* dapat terhubung dengan datacenter, maka data yang dikirimkan oleh *middleware* dapat dipantau secara *realtime* melalui web-app dengan alamat 10.34.0.35:8000. Pada Gambar 5.4 menunjukkan bahwa data dapat ditampilkan pada web-app. Dengan demikian, pengiriman data dari *publisher* melalu *middleware* hingga ke web-app berjalan sesuai dengan harapan.



Gambar 5.4 Data Web-app

## 5.2 Metode Pengujian

Metode pengujian yang digunakan dalam penelitian ini sama dengan metode pengujian pada penelitian sebelumnya sehingga data yang diperoleh dapat digunakan sebagai perbandingan.

### 5.2.1 Pengujian Penggunaan CPU dan Memori

Untuk mendapatkan data kinerja dari CPU dan Memori pada *middleware* digunakan program sebagai berikut:

```
var usage = require('usage');
var pid = 886
setInterval(function() {
    var options = { keepHistory:true};
    usage.lookup(pid, options, function(err, stat) {
        console.log(err, stat);
        console.log(new Date().toISOString());
    });
}, 15000);
```

Kode Program 5.1 Penggunaan CPU dan Memori Raspberry Pi Zero

Program tersebut akan mencatat log penggunaan CPU dan memori berdasarkan PID (*process ID*) dari *middleware* setiap 15 detik sekali yang dilakukan selama 3 jam. Data dari penggunaan CPU dan memori disimpan dalam log file dengan format .txt.

### 5.2.2 Pengujian *Delay* dan Data Transfer

Pada pengujian ini digunakan program tcpdump untuk menangkap lalu lintas data pada *interface* wlan0 (Kode Program 5.2) dan eth0 (Kode Program 5.3) yang kemudian dilakukan analisis lebih lanjut dengan menggunakan program wireshark. Nilai *delay* didapatkan dengan cara mengurangi waktu paket saat diterima dengan waktu paket saat dikirimkan. Untuk mendapatkan nilai end-to-end *delay* adalah dengan menjumlahkan *delay* dari NodeMCU ke *middleware* dan *delay* dari *middleware* ke aplikasi. Data transfer adalah berapa banyak data yang berhasil dikirim ke *middleware* oleh *publisher* dan ke aplikasi oleh *middleware*.

```
tcpdump -i wlan0 -w <nama-file>
```

**Kode Program 5.2 Tcpdump wlan0**

```
tcpdump -i eth0 -w <nama-file>
```

**Kode Program 5.3 Tcpdump eth0**

### 5.2.3 Pengujian Skalabilitas

Pada pengujian skalabilitas digunakan package async dari npm yang dapat menjalankan program secara asynchronous. Program pengiriman data (*publish*) dan program meminta data (*subscribe*) dimasukkan kedalam program async seperti pada Kode Program 5.4. Pada pengujian ini ditentukan jumlah *publisher* dan *subscriber* dengan variasi 100, 500, 1000.

```
async.times (1000, function(n, next) {  
    Publisher ()  
    {  
        next ();  
    };  
}, function (err) {  
});
```

**Kode Program 5.4 Skalabilitas Raspberry Pi Zero**

## 5.3 Skenario Pengujian

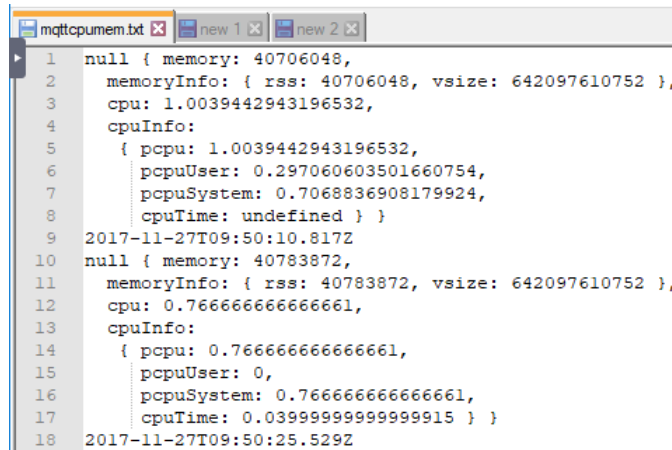
Dalam penelitian ini dilakukan 5 skenario pengujian yaitu:

1. Skenario 1 untuk pengujian penggunaan CPU dan memori adalah dengan menggunakan 1 nodeMCU MQTT yang akan dilakukan selama 3 jam.
2. Skenario 2 untuk pengujian penggunaan CPU dan memori adalah dengan menggunakan 1 nodeMCU CoAP yang akan dilakukan selama 3 jam.

3. Skenario 3 untuk pengujian penggunaan CPU dan memori adalah dengan menggunakan 1 nodeMCU MQTT dan 1 nodeMCU CoAP yang akan dilakukan selama 3 jam.
4. Skenario 4 untuk pengujian *delay* dan data transfer adalah dengan menggunakan 5 nodeMCU MQTT dan 5 nodeMCU CoAP yang mengirim data ke *middleware* secara bersamaan yang akan dilakukan selama 1 jam.
5. Skenario 5 untuk pengujian skalabilitas adalah dengan menjalankan program async dengan variasi *publisher*/susbscriber sebanyak 100, 500, dan 1000.

### 5.3.1 Pengujian Skenario 1

Pada pengujian skenario 1 ini dilakukan pengujian dengan menggunakan 1 nodeMCU MQTT dengan durasi pengujian selama 3 jam untuk mengetahui bagaimana kinerja penggunaan cpu dan memori dari *middleware* jika menggunakan raspberry pi zero.



```

1 null { memory: 40706048,
2   memoryInfo: { rss: 40706048, vsize: 642097610752 },
3   cpu: 1.0039442943196532,
4   cpuInfo:
5     { pcpu: 1.0039442943196532,
6       pcpuUser: 0.297060603501660754,
7       pcpuSystem: 0.7068836908179924,
8       cpuTime: undefined } }
9 2017-11-27T09:50:10.817Z
10 null { memory: 40783872,
11   memoryInfo: { rss: 40783872, vsize: 642097610752 },
12   cpu: 0.7666666666666661,
13   cpuInfo:
14     { pcpu: 0.7666666666666661,
15       pcpuUser: 0,
16       pcpuSystem: 0.7666666666666661,
17       cpuTime: 0.03999999999999915 } }
18 2017-11-27T09:50:25.529Z

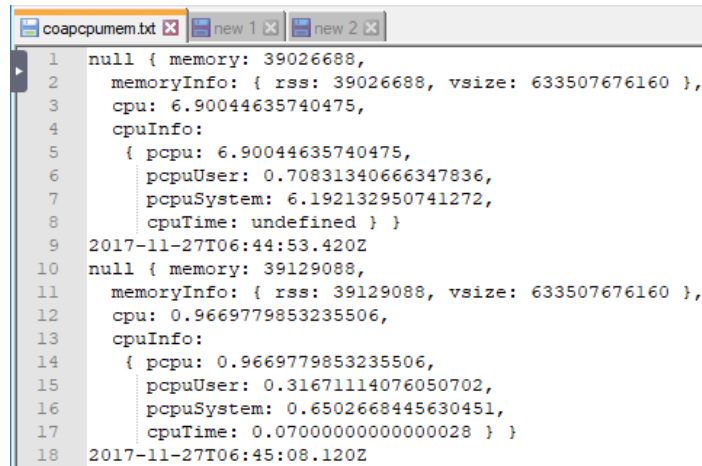
```

**Gambar 5.5 Pengujian Skenario 1**

Gambar 5.5 menunjukkan cuplikan hasil log penggunaan cpu dan memori selama pengujian skenario 1 berlangsung. Log ini merupakan output dari program monitoring cpu dan memori pada bab sebelumnya. Tujuan dari monitoring adalah untuk mengetahui seberapa besar beban penggunaan cpu dan memori.

### 5.3.2 Pengujian Skenario 2

Pada pengujian skenario 2 ini dilakukan pengujian dengan menggunakan 1 nodeMCU CoAP dengan durasi pengujian selama 3 jam untuk mengetahui bagaimana kinerja penggunaan cpu dan memori dari *middleware* jika menggunakan raspberry pi zero.



```

1 null { memory: 39026688,
2   memoryInfo: { rss: 39026688, vsize: 633507676160 },
3   cpu: 6.90044635740475,
4   cpuInfo:
5     { pcpu: 6.90044635740475,
6       pcpuUser: 0.70831340666347836,
7       pcpuSystem: 6.192132950741272,
8       cpuTime: undefined } }
9 2017-11-27T06:44:53.420Z
10 null { memory: 39129088,
11   memoryInfo: { rss: 39129088, vsize: 633507676160 },
12   cpu: 0.9669779853235506,
13   cpuInfo:
14     { pcpu: 0.9669779853235506,
15       pcpuUser: 0.31671114076050702,
16       pcpuSystem: 0.6502668445630451,
17       cpuTime: 0.070000000000000028 } }
18 2017-11-27T06:45:08.120Z

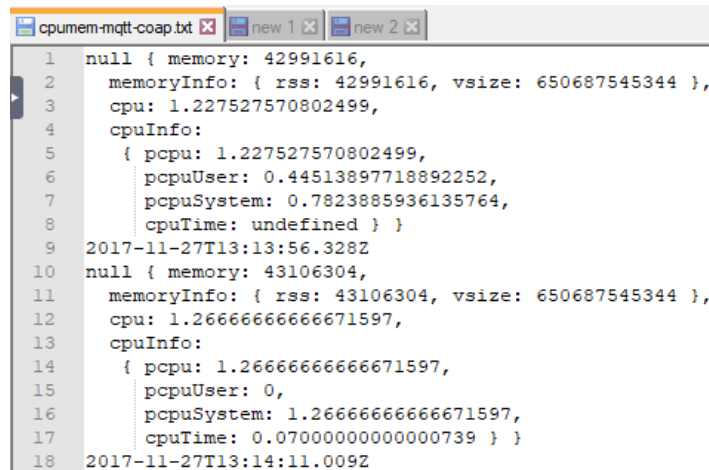
```

**Gambar 5.6 Pengujian Skenario 2**

Gambar 5.6 menunjukkan cuplikan hasil log penggunaan cpu dan memori selama pengujian skenario 2 berlangsung. Log ini merupakan output dari program monitoring cpu dan memori pada bab sebelumnya. Tujuan dari monitoring adalah untuk mengetahui seberapa besar beban penggunaan cpu dan memori.

### 5.3.3 Pengujian Skenario 3

Pada pengujian skenario 3 ini dilakukan pengujian dengan menggunakan 1 nodeMCU MQTT dan 1 nodeMCU CoAP dengan durasi pengujian selama 3 jam untuk mengetahui bagaimana kinerja penggunaan cpu dan memori dari *middleware* jika menggunakan raspberry pi zero.



```

1 null { memory: 42991616,
2   memoryInfo: { rss: 42991616, vsize: 650687545344 },
3   cpu: 1.227527570802499,
4   cpuInfo:
5     { pcpu: 1.227527570802499,
6       pcpuUser: 0.44513897718892252,
7       pcpuSystem: 0.7823885936135764,
8       cpuTime: undefined } }
9 2017-11-27T13:13:56.328Z
10 null { memory: 43106304,
11   memoryInfo: { rss: 43106304, vsize: 650687545344 },
12   cpu: 1.26666666666671597,
13   cpuInfo:
14     { pcpu: 1.26666666666671597,
15       pcpuUser: 0,
16       pcpuSystem: 1.26666666666671597,
17       cpuTime: 0.0700000000000000739 } }
18 2017-11-27T13:14:11.009Z

```

**Gambar 5.7 Pengujian Skenario 3**

Gambar 5.7 menunjukkan cuplikan hasil log penggunaan cpu dan memori selama pengujian skenario 3 berlangsung. Log ini merupakan output dari program monitoring cpu dan memori pada bab sebelumnya. Tujuan dari monitoring adalah untuk mengetahui seberapa besar beban penggunaan cpu dan memori.

#### 5.3.4 Pengujian Skenario 4

Pada pengujian skenario keempat ini, digunakan sebanyak 5 nodeMCU MQTT dan 5 nodeMCU CoAP sebagai *publisher* dengan topik yang berbeda-beda. Mekanisme pengujian ini adalah dengan membuat 10 nodeMCU mengirimkan data secara bersamaan selama 1 jam dan dilakukan capture wireshark pada saat pengujian berlangsung. Pada Tabel 5.1 dan Tabel 5.2 dapat dilihat IP untuk masing-masing protokol MQTT dan CoAP.

**Tabel 5.1 IP Protokol MQTT**

Source	Destination	Protocol	Info
192.168.43.20 (barrack-1)	192.168.43.1	MQTT	<i>Publish Message</i>
192.168.43.24 (barrack-2)	192.168.43.1	MQTT	<i>Publish Message</i>
192.168.43.26 (barrack-3)	192.168.43.1	MQTT	<i>Publish Message</i>
192.168.43.31 (barrack-4)	192.168.43.1	MQTT	<i>Publish Message</i>
192.168.43.29 (barrack-5)	192.168.43.1	MQTT	<i>Publish Message</i>

**Tabel 5.2 IP Protokol CoAP**

Source	Destination	Protocol	Info
192.168.43.21 (kitchen-1)	192.168.43.1	CoAP	CON, MID:23346, POST, TKN:6e 6f 64 65, coap://192.168.43.1/r/home/kitchen
192.168.43.25 (kitchen-2)	192.168.43.1	CoAP	CON, MID:24031, POST, TKN:6e 6f 64 65, coap://192.168.43.1/r/home/kitchen
192.168.43.28 (kitchen-3)	192.168.43.1	CoAP	CON, MID:63061, POST, TKN:6e 6f 64 65, coap://192.168.43.1/r/home/kitchen
192.168.43.30 (kitchen-4)	192.168.43.1	CoAP	CON, MID:23327, POST, TKN:6e 6f 64 65, coap://192.168.43.1/r/home/kitchen
192.168.43.31 (kitchen-5)	192.168.43.1	CoAP	CON, MID:63083, POST, TKN:6e 6f 64 65, coap://192.168.43.1/r/home/kitchen

#### 5.3.5 Pengujian Skenario 5

Pengujian skenario 5 adalah pengujian skalabilitas yang dilakukan untuk mengetahui kemampuan *middleware* dalam menangani peningkatan jumlah *publisher* ataupun *subscriber*. Dalam pengujian ini jumlah *publisher* dan *subscriber* yang digunakan adalah 100, 500, 1000. Hasil dari log pengujian skalabilitas seperti pada Gambar 5.8.

```

953 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
954 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
955 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
956 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
957 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
958 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
959 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
960 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
961 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
962 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
963 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
964 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
965 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
966 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
967 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
968 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
969 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
970 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
971 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
972 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack
973 0|qoap | 7/4/2017 3:54:04 PM 650 [36mMQTT 650 [39m - Client mqttjs_f3a2d8be publish a message to home/barrack

```

**Gambar 5.8 Log Skalabilitas**

Pengujian skalabilitas dilakukan dengan menggunakan program skalabilitas dengan package *async* dari npm dan dilakukan perhitungan berapa banyak *publisher* atau *subscriber* yang dapat ditangani oleh *middleware* dalam satu detik.

## 5.4 Pengambilan Data

### 5.4.1 Pengambilan Data CPU dan Memori

```

null { memory: 42991616,
  memoryInfo: { rss: 42991616, vsize: 650687545344 },
  cpu: 1.227527570802499,
  cpuInfo:
    { pcpu: 1.227527570802499,
      pcpuUser: 0.44513897718892252,
      pcpuSystem: 0.7823885936135764,
      cpuTime: undefined } }

```

**Gambar 5.9 Log CPU dan Memori**

Proses pengambilan data log cpu dan memori dicatat setiap 15 detik sekali berdasarkan program yang telah dibuat. Pada Gambar 5.9 data yang didapat adalah data dari penggunaan memori dalam satuan *byte* yang nantinya dibagi dengan jumlah memori pada raspberry pi zero yaitu 455200768 *byte* dan data penggunaan cpu dalam satuan persentase (%). Pencatatan log dilakukan selama 3 jam untuk skenario 1, skenario 2, dan skenario 3. Setelah pencatatan log selesai dilakukan pengumpulan data dan dihitung rata-rata penggunaan cpu dan memori untuk masing-masing skenario.

```

pi@raspberrypi-zero:~$ free -b
              total        used        free      shared    buffers     cached
Mem:      455200768    87035904   368164864    4653056    10100736    40783872
-/+ buffers/cache:    36151296   419049472
Swap:      104853504         0    104853504

```

**Gambar 5.10 Jumlah Memori (byte) Raspberry Pi Zero**

### 5.4.2 Pengambilan Data *Delay* dan Data Transfer

Untuk mendapatkan nilai dari *delay* pada penelitian ini adalah dengan mengurangi waktu pada saat paket diterima dengan waktu pada saat paket dikirimkan. Waktu paket diterima dapat dilihat pada wireshark seperti pada



Gambar 5.11 pada bagian kolom *time*. Sedangkan untuk waktu pengiriman dapat dilihat pada info masing-masing paket yang terdapat keterangan *timestamp*. Sebagai contoh pada Gambar 5.11 waktu pengiriman adalah 1513941876.168959 (format epoch *time*) dan waktu paket diterima adalah 1513941876.218781. Jadi *delay* yang didapat adalah  $1513941876.218781 - 1513941876.168959 = 0.049822$  detik. Dan akan dilakukan perhitungan *delay* untuk setiap paket yang dikirimkan oleh nodeMCU yang kemudian akan dihitung nilai rata-rata dari *delay*. Cara yang sama juga dilakukan untuk menghitung *delay* dari *middleware* ke aplikasi. Sehingga untuk menghitung nilai *end-to-end delay* adalah dengan menjumlahkan *delay* dari nodeMCU ke *middleware* dengan *delay* dari *middleware* ke aplikasi.

No.	Time	Source	Destination	Protocol	Length	Info
38	1513941876.218781	192.168.43.29	192.168.43.1	MQTT	323	Publish Message
56	1513941883.509248	192.168.43.24	192.168.43.1	MQTT	321	Publish Message
97	1513941888.315045	192.168.43.20	192.168.43.1	MQTT	322	Publish Message
236	1513941906.228246	192.168.43.29	192.168.43.1	MQTT	323	Publish Message
246	1513941913.516967	192.168.43.24	192.168.43.1	MQTT	321	Publish Message
268	1513941917.545713	192.168.43.27	192.168.43.1	MQTT	323	Publish Message
305	1513941919.100752	192.168.43.26	192.168.43.1	MQTT	322	Publish Message
423	1513941936.218338	192.168.43.29	192.168.43.1	MQTT	323	Publish Message
435	1513941943.516111	192.168.43.24	192.168.43.1	MQTT	321	Publish Message
458	1513941947.549227	192.168.43.27	192.168.43.1	MQTT	323	Publish Message
482	1513941948.321524	192.168.43.20	192.168.43.1	MQTT	322	Publish Message
496	1513941949.059390	192.168.43.26	192.168.43.1	MQTT	322	Publish Message
631	1513941973.515374	192.168.43.24	192.168.43.1	MQTT	321	Publish Message
> Frame 38: 323 bytes on wire (2584 bits), 323 bytes captured (2584 bits) > Ethernet II, Src: Espressi_60:b7:0e (60:01:94:60:b7:0e), Dst: Shenzhen_6b:90:a1 (7c:dd:90:6b:90:a1) > Internet Protocol Version 4, Src: 192.168.43.29, Dst: 192.168.43.1 > Transmission Control Protocol, Src Port: 29989, Dst Port: 1883, Seq: 1, Ack: 1, Len: 269						
0000	7c dd 90 6b 90 a1 60 01	94 60 b7 0e 08 00 45 00	.k..'. ..E.			
0010	01 35 00 7c 00 00 ff 06	e2 d7 c0 a8 2b 1d c0 a8	.5. .... ..+...			
0020	2b 01 75 25 07 5b 00 00	22 fb 50 0b a8 1a 50 18	+.u%.[. .".P...P.			
0030	16 a8 84 47 00 00 32 8a	02 00 0c 68 6f 6d 65 2f	...G..2. ...home/			
0040	62 61 72 72 61 63 6b 00	0a 7b 22 73 65 6e 73 6f	barrack. .{"senso			
0050	72 22 3a 7b 22 6d 6f 64	75 6c 65 22 3a 22 64 68	r":{"mod ule":"dh			
0060	74 31 31 22 2c 22 74 69	70 65 22 3a 22 65 73 70	t11","ti pe":"esp			
0070	38 32 36 36 22 2c 22 69	6e 64 65 78 22 3a 36 33	8266","i ndex":63			
0080	33 38 33 31 38 2c 22 69	70 22 3a 22 31 39 32 2e	38318,"i p":"192.			
0090	31 36 38 2e 34 33 2e 32	39 22 7d 2c 22 70 72 6f	168.43.2 9"},"pro			
00a0	74 6f 63 6f 6c 22 3a 22	6d 71 74 74 22 2c 22 74	tocol":" mqtt","t			
00b0	6f 70 69 63 22 3a 22 68	6f 6d 65 5c 2f 62 61 72	opic":"h ome\bar			
00c0	72 61 63 6b 22 2c 22 68	75 6d 69 64 69 74 79 22	rack","h umidity"			
00d0	3a 7b 22 76 61 6c 75 65	22 3a 38 30 2c 22 75 6e	:{"value ":80,"un			
00e0	69 74 22 3a 22 25 22 7d	2c 22 74 69 6d 65 73 74	it":"%"} ,"timest			
00f0	61 6d 70 22 3a 7b 22 6d	69 63 72 6f 22 3a 31 36	amp":{"m icro":16			
0100	38 39 35 39 2c 22 73 65	63 22 3a 31 35 31 33 39	8959,"se c":"15139			
0110	34 31 38 37 36 7d 2c 22	74 65 6d 70 65 72 61 74	41876}," temperat			
0120	75 72 65 22 3a 7b 22 76	61 6c 75 65 22 3a 32 35	ure":{"v alue":25			

**Gambar 5.11 Log Delay**

Data transfer yang dimaksud dalam penelitian ini adalah seberapa banyak data yang sukses dikirimkan oleh sensor ke *middleware*. Untuk mendapatkan hasil data transfer adalah dengan melakukan filter IP topik tertentu pada log wireshark dan dilakukan perhitungan seberapa banyak paket yang berhasil *publish* seperti pada Gambar 5.12.

mqtt&ip.src==192.168.43.20						
No.	Source	Destination	Protocol	Length	Time delta	Info
2458	192.168.43.20	192.168.43.1	MQTT	324	0.000000000	Publish Message
7436	192.168.43.20	192.168.43.1	MQTT	324	30.006808000	Publish Message
13282	192.168.43.20	192.168.43.1	MQTT	76	35.238990000	Connect Command
18134	192.168.43.20	192.168.43.1	MQTT	324	30.220287000	Publish Message
23053	192.168.43.20	192.168.43.1	MQTT	324	30.005449000	Publish Message
28481	192.168.43.20	192.168.43.1	MQTT	76	32.555175000	Connect Command
33489	192.168.43.20	192.168.43.1	MQTT	324	30.369245000	Publish Message
38437	192.168.43.20	192.168.43.1	MQTT	324	30.000537000	Publish Message
43278	192.168.43.20	192.168.43.1	MQTT	324	29.998377000	Publish Message
48023	192.168.43.20	192.168.43.1	MQTT	324	29.999324000	Publish Message
52959	192.168.43.20	192.168.43.1	MQTT	324	30.000621000	Publish Message
57824	192.168.43.20	192.168.43.1	MQTT	324	30.001615000	Publish Message
62869	192.168.43.20	192.168.43.1	MQTT	324	30.004071000	Publish Message
67827	192.168.43.20	192.168.43.1	MQTT	324	30.003893000	Publish Message
72655	192.168.43.20	192.168.43.1	MQTT	324	29.999534000	Publish Message
77443	192.168.43.20	192.168.43.1	MQTT	324	30.000934000	Publish Message
82369	192.168.43.20	192.168.43.1	MQTT	324	30.102074000	Publish Message
87263	192.168.43.20	192.168.43.1	MQTT	324	29.888518000	Publish Message
92136	192.168.43.20	192.168.43.1	MQTT	324	30.002207000	Publish Message

Gambar 5.12 Log Data Transfer

### 5.4.3 Pengambilan Data Skalabilitas

Data skalabilitas diperoleh dari log file dalam bentuk .txt yang merupakan output dari program skalabilitas. Pada log file tersebut dicatat waktu kapan proses *publish*, *post*, atau *subscribe* dilakukan. Waktu inilah yang digunakan untuk menghitung seberapa banyak *concurrent publish/subscribe* dalam satu detik. Perhitungan dilakukan untuk seluruh variasi jumlah *publish/subscribe* yakni 100, 500, 1000. Contoh log file skalabilitas dengan *publisher coap* berjumlah 100 seperti pada Gambar 5.13.

coap100.txt		Now streaming realtime logs for [all] processes	
71	[STREAMING]	Now streaming realtime logs for [all] processes	
72	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
73	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
74	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
75	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
76	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
77	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
78	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
79	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
80	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
81	l coap	12/19/2017 6:14:27 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
82	l coap	12/19/2017 6:14:32 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
83	l coap	12/19/2017 6:14:32 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
84	l coap	12/19/2017 6:14:32 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
85	l coap	12/19/2017 6:14:32 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
86	l coap	12/19/2017 6:14:32 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
87	l coap	12/19/2017 6:14:32 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
88	l coap	12/19/2017 6:14:32 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen
89	l coap	12/19/2017 6:14:32 PM	650 [32mCOAP650]39m - Incoming POST request from 192.168.43.18 for home/kitchen

Gambar 5.13 Log Data Skalabilitas

## 5.5 Pengolahan Data

### 5.5.1 Pengolahan Data CPU dan Memori

Dari hasil log file program monitoring cpu dan memori yang dilakukan selama 3 jam diperoleh data sebanyak 720 karena program mencatat log penggunaan cpu dan memori setiap 15 detik sekali untuk skenario 1 hingga skenario 3. Dari 720 data tersebut dibagi menjadi tiga bagian masing-masing 240 data yang mana bagian pertama menunjukkan penggunaan cpu dan memori pada satu jam

pertama, bagian kedua menunjukkan penggunaan cpu dan memori pada satu jam kedua, dan bagian ketiga menunjukkan penggunaan cpu dan memori pada satu jam ketiga. Dari hasil pembagian tersebut diperoleh rata-rata penggunaan cpu dan memori mulai dari satu jam pertama hingga satu jam ketiga seperti pada Tabel 5.3 untuk rata-rata penggunaan cpu dan Tabel 5.4 untuk rata-rata penggunaan memori. Data selengkapnya dapat dilihat pada Lampiran A Penggunaan CPU dan Lampiran B Penggunaan Memori.

**Tabel 5.3 Rata-Rata Penggunaan CPU**

	MQTT	CoAP	MQTT+CoAP
1st hour	0,87%	0,91%	1,25%
2nd hour	0,84%	0,88%	1,22%
3rd hour	0,87%	0,84%	1,27%
Average	0,86%	0,88%	1,25%

**Tabel 5.4 Rata-Rata Penggunaan Memori**

	MQTT	CoAP	MQTT+CoAP
1st hour	10,10%	9,63%	12,58%
2nd hour	10,24%	9,73%	12,78%
3rd hour	10,37%	9,89%	12,99%
Average	10,24%	9,75%	12,79%

### 5.5.2 Pengolahan Data *Delay* dan Data Transfer

Data *delay* dan data transfer diperoleh dari hasil *capture traffic* menggunakan program tcpdump. Terdapat dua data *capture traffic*, yang pertama adalah *capture traffic* dari nodeMCU ke *middleware* dan yang kedua adalah *capture traffic* dari *middleware* ke aplikasi/web-app. Data yang diperoleh tersebut dilakukan analisis menggunakan program wireshark. Untuk memperoleh nilai *delay* adalah dengan mengurangi waktu saat paket diterima oleh dengan waktu saat paket dikirimkan. Sedangkan untuk mendapatkan hasil data transfer adalah dengan melakukan filter pada wireshark untuk masing-masing topik berdasarkan IP yang didapat kemudian dihitung berapa banyak data yang berhasil dikirimkan. Hasil yang didapat dari pengolahan data *delay* dan data transfer ditunjukkan oleh Tabel 5.5, Tabel 5.6, dan Tabel 5.7. Data selengkapnya dapat dilihat pada Lampiran C *Delay* dan Data Transfer.

**Tabel 5.5 Data MQTT**

MQTT						
Topic	Expected	Actual	Invalid	Loss Rate	Success rate	Average Delay(s)
barrack-1	120	120	0	0%	100%	0,357

MQTT						
Topic	Expected	Actual	Invalid	Loss Rate	Success rate	Average Delay(s)
barrack-2	120	120	0	0%	100%	0,566
barrack-3	120	120	0	0%	100%	0,123
barrack-4	120	120	0	0%	100%	0,578
barrack-5	120	120	0	0%	100%	0,242
Total	600	600	0	0%	100%	0,373

**Tabel 5.6 Data CoAP**

CoAP						
Topic	Expected	Actual	Invalid	Loss Rate	Success rate	Average Delay (s)
kitchen-1	120	120	0	0%	100%	0,776
kitchen-2	120	120	0	0%	100%	0,647
kitchen-3	120	120	0	0%	100%	0,653
kitchen-4	120	120	0	0%	100%	0,128
kitchen-5	120	120	0	0%	100%	0,131
Total	600	600	0	0%	100%	0,467

**Tabel 5.7 Data Websocket**

Websocket			
Expected	Actual	Overhead	Average Delay (s)
1200	1200	0%	0,495

### 5.5.3 Pengolahan Data Skalabilitas

Pada log file data skalabilitas terdapat log waktu kapan proses *publish/subscribe* dilakukan. Berdasarkan waktu tersebut dapat diketahui berapa banyak *publisher/subscriber* yang dapat ditangani oleh *middleware* dalam satu detik dengan variasi 100, 500, dan 1000 klien. Dilakukan perhitungan berapa banyak data yang memiliki log waktu yang sama sehingga menghasilkan data *concurrent publish/subscribe* dan menghitung jeda waktu saat proses *publish/subscribe* dimulai hingga waktu proses *publish/subscribe* selesai sehingga menghasilkan data *time publish/subscribe*.

**Tabel 5.8 Skalabilitas CoAP**

CoAP			
	100 Clients	500 Clients	1000 Clients
<i>Publisher</i>	41	35	39
<i>Time Publish (s)</i>	6	36	72

**Tabel 5.9 Skalabilitas MQTT**

MQTT			
	100 Clients	500 Clients	1000 Clients
<i>Publisher</i>	81	55	56
<i>Time Publish (s)</i>	3	10	20

**Tabel 5.10 Skalabilitas Websocket**

Websocket			
	100 Clients	500 Clients	1000 Clients
<i>Subscriber</i>	50	52	54
<i>Time Subscribe (s)</i>	3	12	22

Pada Tabel 5.8, Tabel 5.9, Tabel 5.10 perhitungan *publisher/subscriber* diambil berdasarkan jumlah maksimal *publisher/subscriber* dalam satu detik yang terdapat dalam log file skalabilitas. Sedangkan *time publish/subscribe* adalah jeda waktu saat proses *publish/subscribe* dimulai hingga waktu proses *publish/subscribe* selesai. Data selengkapnya dapat dilihat pada Lampiran D Skalabilitas.